IN THE SPECIFICATION:

Please find attached the corrections to be made to the cited areas of the original specification.

At page 1, line 23, change the paragraph to read as follows:

The mainframe environment was one where a large data processing resource was interconnected to a large number of "dumb terminals". The dumb terminals had minimal computational capabilities and were effectively utilized to display results on a screen and to input data from a keyboard. The terminal [being] <u>was</u> primarily under control of the mainframe computer system which carried out almost all the computational activity.

At page 11 lines 12-14, thru the top of page 12, change the paragraph to read as follows:

Currently all LINC systems are generated by the host-based LINC (133) Fig. 18 Development Environment (LDE) from specifications kept in its repository. The specifications [[(113)]] (133) Fig. 18 can be entered directly into the repository using the host-based editor and painter (134) or they can be prepared in the PC-based LDA environment (Fig. 18, 130) and loaded in using the CaseLoad utility (131) via an LDE repository in Lcopy format and loaded back in to the same or another host-based repository.

At page 12, lines 2 and 9, change the paragraph to read as follows:

The generate process (135) Fig. 18 takes a specification and determines what compilations and database reorganizations are needed to transform the existing runtime system and its database so that it matches the new specification. Moving from a fully functional version of a system to the next version is fully automated, the user just generates the system and when the generate process is finished, [[the]] they once again have a fully functional system.

At page 12, line 13, change the paragraph to read as follows:

A generated specification can be transferred to another environment without regenerating it again. This facility can be used, for example, to move a system from a test environment into production. These transferred systems can be configured to match their new environment.

At page 12, line 21, change the paragraph to read as follows:

LINC also provides a number of tools and facilities. Some of these let the system administrator manage and maintain a running LINC system. Other facilities are used at runtime to supplement system software and provide a necessary functionality. Amongst the tools and other facilities that are provided are:

At page 13, line 6, change the paragraph to read as follows:

LDA: (FIG. 19)

At page 14, line 21, change the paragraph to read as follows:

In the preferred embodiment, a system is provided for readily extending the utility of a 4GL application into a web environment. The extension is achieved by means of taking the originally defined 4GL application and providing a translator to create a series of codes, the codes being constructed in the language Java. Reference is made to the standard texts such as the Sunsoft Press Java Series of texts for information on the Java language. The codes provide the functionality for interaction with the previously generated 4GL application over the internet. In this manner, internet interaction can be achieved with a previously constructed host application in a substantially simplified and automated manner.

At page 14, line 31, change the paragraph to read as follows:

The following description will now be given in relation to an embodiment designed for the LINC System and employing LINC terminology. It [[would]] <u>should be</u> appreciated, however, that the invention can extend to any 4GL application.

At page 15, line 22, thru the top of page 16, change the paragraph to read as follows:

The generated client side 20 of ActiveLINC is preferably written entirely in Java and consists of a runtime environment and generated Ispec components 22 and, optionally, generated GUI interface applications 21. The generated Ispec components can be used as JavaBeans or ActiveX components, so providing access to LINC application from customer-written client applications such as Microsoft Word and Excel. As the Active LINC code is written in Java, it is capable of running on any qualified Java 1.1 platform, including Apple Macintosh, UNIX workstations, Windows PCs, OS/2 PCs and Network Computers.

At page 16, line 18, change the paragraph to read as follows:

Each client application can run independently of all other clients, with its own connection back to the application server. Preferably, there are not points of single threading or possible congestion in the transaction path to add overhead and reduce performance. RATL servers can be designed to offer connectivity, throughput and transit times comparable to terminal access on the same host systems. The RATL protocol used between client and application servers only transfers transaction data and responses and adds minimal overhead to each message. Forms and screen layouts in interface 21 can be translated to Java code at code generation time and the compiled code can be loaded to the client only once. Form definitions and layouts are not sent down to the client with every transaction response and there is no run-time interpretation of form layouts.

At page 18, line 29, thru the top of page 19, change the paragraph to read as follows:

Turning now to Fig. 3, there is illustrated an example ActiveLINC application in more detail. The user interface 21 and interface components 22 are initially stored on a web server 30 and are utilized on demand as is the normal case of Java applets loaded via a browser. The interface components 22 interact with a LINC application 23 and include components 31 for handling Ispec requests in addition to a console handler 32 dealing with console requests, a message handler 33 for dealing with messages, an "Ispec factory" 34 for handling Ispec updates, a transaction manager 35 for dealing with an overall transaction and a list box manager 36 for dealing with an overall transaction and a list box manager 36 for handling list boxes 37. The interface components 22 are utilized by a created graphical user interface application 21 which can include separate views for a console view 40, an error view window 41 for handling error messages, a series of Ispec views 42-44 corresponding to various screen views for customer, in addition to a number of corresponding Ispec objects 46-48 which can comprise data objects for forwarding to interface components manager 22.

At page 21, line 8, change the paragraph to read as follows:

Examining the component interface in more detail ActiveLINC client applications access the remote LINC application through a set of objects. As illustrated in Fig. 5, the primary object is the LINC Environment's object 60 which represents a connection to a remote LINC application. This object is used to create Ispec objects e.g. 61 from the generated Ispec components 62 and to handle the exchange of transaction requests and responses with the remote LINC application. Client applications can have multiple LINC Environment objects, giving them concurrent access to multiple LINC applications, possibly running on different host systems.

At page 24, line 15, change the paragraph to read as follows:

Everything needed to support GUI [[form]] <u>for</u> access to LINC applications is provided by the LINC Environment and Ispec components and the generated interface applications use all of this functionality. Client applications can choose to use just the simplest interface methods or they can take full advantage of list boxes and consoles for running reports and existing navigational logic in their LINC application.

At page 27, lines 7-9, change the paragraph to read as follows:

3. getFields returns array of String

returns the names of the fields defined in the Ispec, excluding copy.from fields, as an array of strings. Methods are available to find the properties of each of these[[s]] named fields, such as type and length. The names of [[copyfrom]] copy.from fields can be fetched by using getArrayColumnNames which returns an array of strings. The number of copy.from rows and columns can also be obtained by calling the appropriate introspection methods (getArrayRows and getArrayCols).

At page 33, line 25 (though the top of page 34), change the paragraph to read as follows:

ActiveLINC client applications that are run as Java applets from a Web browser may not always be able to [[n]] make a direct connection back to the LINC application server because of standard Java web security constraints. The security constraints may be such that the Java applets may only open TCP/IP connections back to the Web server from where the applets were themselves loaded. This is not a problem if the Web server providing the applets and the LINC application are running on the same system. Fig. 8 shows an ActiveLINC client applet running from a Web browser and accessing a remote LINC application. In this case the Web server is running on the same system as the RATL server and LINC applications. This configuration is possible with Windows NT and UNIX systems, and with the Unisys WebTS and Atlas Web servers running on ClearPath systems. A variant on this configuration would be to have restricted Web servers running on the LINC application servers that just supply the ActiveLINC applets. This would both support load partitioning and make system maintenance easier as all the generated LINC and ActiveLINC code could be loaded onto the same system.

At page 34, line 35, (through the top of page 35), change the paragraph to read as follows:

ActiveLINC clients and RATL servers normally exchange unencrypted messages that are susceptible to eavesdropping, especially over a public network. Secure communications is possible through use of 'secure sockets'(SSL) instead of standard TCP/IP connections between ActiveLINC clients and the application servers. SSI, is not yet available on MCP and [[OS22200]] OS2200 systems but secure data transport can still be provided through the use of a redirector as shown in Fig. 10. Connections can be used between the ActiveLINC client and an SSL-capable Web server and normal TCP/IP is used to connect to the application servers over a physically secure local network.

At page 37, line 5, change the paragraph to read as follows:

ActiveLINC interface programs offer good performance and scalability. The Ispec form definitions are translated to Java code at ActiveLINC generation time to run as normal Java GUI programs. Form [[artefacts]] artifacts, such as display fields, are painted on to the user's screen by direct calls on to the standard Java graphics package. There is no run-time overhead in interpreting screen layouts. Separating out form descriptions from transaction data also improves performance. Form programs are downloaded at most once per session rather than for each form. Only transaction data is sent between the clients and servers on each exchange. The direct connections between clients and servers also adds to the good performance of ActiveLINC clients by avoiding placing heavy processing loads on Web servers, such as dynamically creating HTML Web pages.